Distributed vs. Centralized Particle Swarm Optimization for Learning Flocking Behaviors

Iñaki Navarro, Ezequiel Di Mario and Alcherio Martinoli

Distributed Intelligent Systems and Algorithms Laboratory,
School of Architecture, Civil and Environmental Engineering,
École Polytechnique Fédérale de Lausanne
{ezequiel.dimario, inaki.navarro, alcherio.martinoli}@epfl.ch

Abstract

In this paper we address the automatic synthesis of controllers for the coordinated movement of multiple mobile robots. We use a noise-resistant version of Particle Swarm Optimization to learn in simulation a set of 50 weights of a plastic artificial neural network. Two learning strategies are applied: homogeneous centralized learning, in which every robot runs the same controller and the performance is evaluated externally with a global metric, and heterogeneous distributed learning, in which robots run different controllers and the performance is evaluated independently on each robot with a local metric. The two sets of metrics enforce Reynolds' flocking rules, resulting in a good correspondence between the metrics and the flocking behaviors obtained. Results demonstrate that it is possible to learn the collective task using both learning approaches. The solutions from the centralized learning have higher fitness and lower standard deviation than those learned in a distributed manner. We test the learned controllers in real robot experiments and also show in simulation the performance of the controllers with increasing number of robots.

Introduction

This article tackles the synthesis of high-dimensional controllers for cooperative tasks performed by resource-constrained robots. Evaluative machine-learning techniques are an alternative to model-based control design that may allow for full exploitation of the platforms' limited sensing capabilities, coping with discontinuities and nonlinearities, as well as dealing with noise in the performance evaluations (Floreano and Mondada, 1996; Baldassarre et al., 2007; Gauci et al., 2014; Jin and Branke, 2005; Pugh and Martinoli, 2009).

As in our previous work (Di Mario et al., 2014b), the cooperative task chosen is a loosely-coordinated collective movement or flocking (Balch and Arkin, 1998; Olfati-Saber, 2006; Antonelli et al., 2008; Navarro and Matía, 2011), in which a group of robots move together. Some researchers have previously shown that it is feasible to use learning to generate cooperative behaviors (Matarić, 2001; Parker, 1997; Baldassarre et al., 2007; Gauci et al., 2014). Matarić (2001) and Parker (1997) addressed the topic of learning in multi-robot teams using a small number of parameters per robot, as opposed to the large search space considered in this paper. It should be noted that the task as implemented in this article is harder than those presented in other contributions as the robots are not physically connected to each other (Baldassarre et al., 2007), they are required not only to aggregate but also move together (Gauci et al., 2014), and there is no environmental template or goal to guide their movement (Floreano and Mondada, 1996). Finally, in the case of Baldassarre et al. (2007) and Gauci et al. (2014) learning has been done only in a centralized manner, using homogeneous controllers and a global performance metric.

Morihiro et al. (2006) used Q-learning to generate flocking behaviors of virtual agents (not robots) in the presence of a predator, where the agents individually learn discrete actions similar to Reynolds' rules.

Some researchers have used different optimization techniques to improve the performance of manually designed flocking controllers, using PSO (Lee and Myung, 2013; Etemadi et al., 2012), gradient descent (Chang et al., 2013), Reinforcement Learning (Hayes and Dormiani-Tabatabaei, 2002), or Evolutionary Strategies (Celikkanat, 2008). Our approach in this article differs in that our behaviors are generated by a highly plastic artificial neural network and not by a specific control design targeted to flocking behavior. In other words, the main goal of this article is to compare centralized and distributed learning methods for design and optimization of collaborative behaviors, among which flocking has been chosen as a benchmark.

The distributed learning evaluates several candidate solutions in parallel on the available robotic resources. Such an approach allows the distributed robotic system to increase its robustness to failure of individual robots and speed up the overall learning process (Di Mario and Martinoli, 2014b). In order to compare the distributed and centralized approaches, we aim to design a pair of global and local fitness functions that result in the desired flocking behavior. The local or individual metric must be evaluated locally by each robot and be close to the global metric.

The second aim of this paper is to get additional correspondence between the fitness metric used and the flocking

Figure 1: Four Khepera III robots performing one of the learned flocking algorithms presented in this article.

behavior observed, in particular in respect to our previous work (Di Mario et al., 2014b). In order to achieve it, we augmented the fitness metrics to enforce the three Reynolds' flocking rules (Reynolds, 1987), by adding alignment with neighboring flockmates to the originally implemented collision avoidance and attraction. As a consequence of a better alignment and therefore tighter motion coordination, the local and global performances match better.

The remainder of this article is organized as follows. In the next section we describe the robotic platform, learning algorithms, fitness metrics and control architecture. In Section Experimental Results and Discussion, we present the different experiments performed and discuss the results obtained both in simulation and with real robots. Finally, the last section draws the conclusions of this work and discusses the limitations of the approach.

Methodology

A variation of Particle Swarm Optimization (PSO) (Kennedy and Eberhart, 1995) is used in this article in order to learn flocking behaviors. The learning problem for PSO is choosing a set of parameters of an underlying robotic controller such that a given fitness metric is maximized. The learning process is performed completely in simulation, while the learned solutions are tested both using high-fidelity simulation and real robots.

Experimental Platform

The experimental platform used is the Khepera III mobile robot, a differential wheeled vehicle with a diameter of 12 cm (see Fig. 1). Its sensing capabilities are augmented with a relative positioning system (Pugh et al., 2009), which calculates range and bearing to nearby robots based on the strength of an infrared signal. The system also communicates the ID of the robot, allowing to estimate also the heading of neighboring robots by exchanging the bearings between a pair of robots. In our experiments this communication is done using the IEEE 802.11 wireless standard and UDP messages. The Khepera III has two wheel encoders, which are used to estimate the trajectory followed by the robots for the local fitness calculations.

Simulations are performed in Webots (Michel, 2004), a high-fidelity submicroscopic simulator that models dynamical effects such as friction and inertia. In this context, by

```
Initialize particles
for N_i iterations do
    for N_p particles do
        Update particle position
        Evaluate particle
        Re-evaluate personal best
        Aggregate with previous best
        Share personal best
    end for
end for
```

Figure 2: Noise-resistant PSO algorithm.

submicroscopic we mean that it provides a higher level of detail than usual microscopic models, faithfully reproducing intra-robot modules (e.g., individual sensors and actuators). The simulator has a built-in relative positioning system that gives information about the distance and direction to neighboring robots within line-of-sight, mimicking the one used in the real robots.

Learning Algorithm

The PSO algorithm used is a noise-resistant version introduced by Pugh et al. (2005). It works by re-evaluating personal best positions and aggregating them with the previous evaluations, in our case by performing a regular average at each iteration of the algorithm. The pseudocode for the algorithm is presented in Fig. 2.

Each particle position represents a set of parameters of the controller. As defined in Eq. 1, the movement of particle i in dimension j depends on three components: the velocity at the previous step weighted by an inertia coefficient w, a randomized attraction to its personal best $x^*_{i,j}$ weighted by w_p, and a randomized attraction to the neighborhood's best $x^*_{i',j}$ weighted by w_n. $rand()$ is a random number drawn from a uniform distribution between 0 and 1.

$$v_{i,j} = w \cdot v_{i,j} + w_p \cdot rand() \cdot (x^*_{i,j} - x_{i,j}) + w_n \cdot rand() \cdot (x^*_{i',j} - x_{i,j}) \quad (1)$$

Using the PSO algorithm we explore two different learning schemes, characterized by the way the particles are distributed among the robots and the fitness function used. The first, *global homogeneous*, copies the same candidate solution (or set of weights) to every robot, and uses a global fitness function that evaluates the group behavior. The second, *local heterogeneous*, distributes a different candidate solution to each robot, and uses a local fitness function that is evaluated independently and individually on each robot. The distributed version allows to speed up the evaluations by a factor equal to the number of robots.

The PSO neighborhood is implemented as a ring topology with one neighbor on each side. Particles' positions and velocities are initialized randomly with a uniform distribution in the $[-20, 20]$ interval, and their maximum velocity

Table 1: PSO parameter values

Parameter	Value
Number of robots N_{rob}	4
Swarm size N_p	52
Iterations N_i	200
Evaluation span t_e	4x45 s
Re-evaluations N_{re}	1
Personal weight w_p	2.0
Neighborhood weight w_p	2.0
Dimension D	50
Inertia w	0.8
V_{max}	20

is also limited to that interval. The PSO algorithmic parameters (see Table 1) are set following the guidelines for limited-time adaptation presented in our previous work (Di Mario and Martinoli, 2014a). These guidelines recommend a swarm size equal to the dimension of the search space. Since the dimension is 50 and four robots are used, we round up the swarm size to 52 particles in order to have exactly 13 particles per robot in the distributed implementation.

It is worth noticing that in the distributed heterogeneous learning, groups of four particles are always evaluated together as a flock of four robots. In these groups of four particles, two of them have a PSO neighborhood of particles from the same group, while each of the other two share their neighborhood with one particle of another group and one from its group. When testing the best controller from a distributed heterogeneous learning, we find the particle with the best local performance and test it together with the other three particles of its group.

Fitness Functions

In this section, we define the fitness functions used for centralized and distributed learning, using a global metric for the first and a local metric for the second. Both performance functions have three factors: movement, alignment, and compactness. These factors reward robots that move as far as possible from their initial positions, align their headings, and stay close to each other without colliding. The factors are all normalized to the interval $[0,1]$.

In the real experiments, all positions and distances used in the global performance metric are obtained with a global tracking system that can detect all robots at any given time, using an overhead camera connected to a computer running SwisTrack (Lochmatter et al., 2008).

The movement factor of the global performance metric (f_{1g}) is the normalized distance between the initial and the final positions of the center of mass of the group of robots. The normalization factor is the maximum distance that a robots can travel in one evaluation, i.e., the robot's maximum speed multiplied by the evaluation time.

$$f_{1g} = \frac{|\vec{x_c}(t_f) - \vec{x_c}(t_0)|}{D_{max}} \qquad (2)$$

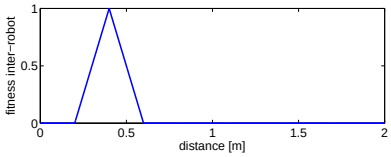


Figure 3: Inter-robot fitness as a function of the distance between two robots.

The global alignment factor (f_{2g}) quantifies the heading difference between two robots (H_{diff}) averaged between every pair of robots and during the evaluation time. It has a maximum value of 1 when all the robots are aligned and tends to 0 when robots are not aligned. It is defined as:

$$f_{2g} = 1 - \frac{1}{N_{eval}} \sum_{k=1}^{N_{eval}} \left(\frac{1}{N_{pairs}} \sum_{j=1}^{N_{pairs}} abs(H_{diff_{j,k}})/\pi\right) \qquad (3)$$

where N_{eval} is the number of time steps in the evaluation period, N_{pairs} is number of inter-robot pairs and $H_{diff_{j,k}}$ is the difference of heading between pair j at time step k. Note that if there are more than two robots its value can never be 0.

The global compactness factor (f_{3g}) is the average over the evaluation time and over each pair of robots of the inter-robot fitness. We define the inter-robot fitness between two robots as a function of the distance between them, as shown in Fig. 3. The fitness is maximum at the desired inter-robot distance of $0.4\,m$, and it is zero when the robots are closer than $0.2\,m$ (slightly larger than the robots' diameter) or further apart than $0.6\,m$. It rewards robots that stay close to each other without colliding, implementing two of the Reynolds' rules. At each time step, we calculate the inter-robot fitness for each pair of robots, and then average across all pairs:

$$f_{3g} = \frac{1}{N_{eval}} \sum_{k=1}^{N_{eval}} \left(\frac{1}{N_{pairs}} \sum_{j=1}^{N_{pairs}} fit_{inter_{j,k}}\right) \qquad (4)$$

where $fit_{inter_{j,k}}$ is the inter-robot fitness for inter-robot pair j at time step k.

The local performance metric is calculated individually by each robot, using exclusively on-board resources and mimicking the global metric. The local movement factor (f_{1l}) is defined in two different variations. The first is the normalized distance traveled by the robot (f_{1al}), based on the final position, which is calculated with odometry using the wheel encoders. The second is the normalized distance traveled by the center of mass of the group of robots (f_{1bl}), calculated using the odometry of the robot and the relative position to neighboring robots. If a neighboring robot position can not be estimated (due to occlusions or limited range of the relative positioning system), the last absolute position where the robot was seen is used as final position.

$$f_{1al} = \frac{|\vec{x_i}(t_f) - \vec{x_i}(t_0)|}{D_{max}} \qquad (5)$$

$$f_{1bl} = \frac{|\vec{x}_c(t_f) - \vec{x}_c(t_0)|}{D_{max}} \quad (6)$$

f_{1bl} matches the global movement factor better than f_{1al}, but tends to evaluate all the particles in the group of robots with a very similar performance, although the controllers could be very different. Two different local metrics are employed on this article depending on the local movement factor used.

The local alignment factor (f_{2l}) is equivalent to the global alignment factor measuring the absolute heading difference between pairs of robots as in Eq. 3. The difference here is that each robot calculates its own metric only measuring the heading difference between itself and the other three robots, using the relative positioning system and communication. These measurements might be affected by occlusions and range limitations. If for a time step no neighbor is seen then H_{diff} is set to 1 for that time instant.

The local compactness factor (f_{3l}) is implemented as in Eq. 4, based on the inter-robot fitness. However, in the local metric the number of pairs N_{pairs} in Eq. 4 is modified so that each robot only measures the distance to the other three using the relative positioning system and then averages the inter-robot fitness only for those other three robots, as opposed to averaging across all pairs of robots. Another difference between the local and global compactness factors is that the local inter-robot distance measurements are affected by occlusion, while the global ones are not.

Both global and local fitness are obtained by aggregating the three corresponding factors using a generalized aggregation function described by Zhang et al. (2008):

$$F = \left(\frac{\omega_1 f_1^s + \omega_2 f_2^s + \omega_3 f_3^s}{\omega_1 + \omega_2 + \omega_3} \right)^{\frac{1}{s}} \quad (7)$$

where f_i are the individual fitness factors (with $f_i = f_{il}$ for the local fitness, and $f_i = f_{ig}$ for the global), ω_i their corresponding aggregation weights, and s is the degree of compensation. We set $s = 0$, i.e., the highest degree of compensation in design-appropriate aggregation functions, simplifying Eq. 7 to:

$$F = \lim_{s \to 0} \left(\frac{\omega_1 f_1^s + \omega_2 f_2^s + \omega_3 f_3^s}{\omega_1 + \omega_2 + \omega_3} \right)^{\frac{1}{s}} = (f_1^{\omega_1} f_2^{\omega_2} f_3^{\omega_3})^{\frac{1}{\omega_1 + \omega_2 + \omega_3}} \quad (8)$$

Since the three factors (f_i) are in the interval $[0,1]$, the fitness function F will also be in the same range. The aggregation weights used are: $\omega_1 = 0.4$, $\omega_2 = 0.5$, and $\omega_3 = 0.1$.

In our previous work (Di Mario et al., 2014a), we showed that the fitness evaluations for learning a simpler robotic task had a large standard deviation, and that performing re-evaluations was an effective way of dealing with this challenge in the learning. Given the more complex behavior to be learned in this article and the difficulties encountered while doing so, we decided to perform multiple internal evaluations of the fitness and average them in order to make the learning more robust. Concretely, each candidate solution is evaluated four times during $45\,s$ and its performance averaged ($F' = \frac{1}{4}\sum_{i=1}^{4} F_i$) before consideration by the noise-resistant algorithm shown in Fig. 2.

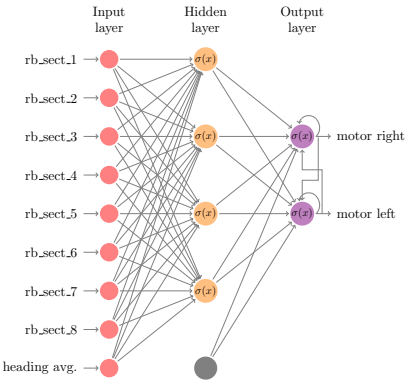


Figure 4: Diagram of the neural network controller. In red are the inputs, yellow the hidden layer with sigmoidal outputs, in blue the sigmoidal outputs which control the motor speed, and in gray the bias input.

Controller Architecture

The controller is an artificial neural network with nine inputs, a hidden layer of four units with sigmoidal activation functions, and two output units also with sigmoidal activation (see Fig. 4). The output neurons have also as input a connection from a constant bias speed, a recurrent connection from its own output, and a lateral connection from the other neuron's output. The controller uses only local, onboard measurements regardless of the performance metric. Its inputs are the range and bearing measurements and the heading average among the robots, while the outputs determine the two wheel speeds. The total number of weights to be optimized by the PSO algorithm is 50. The hidden layer, not present in our previous work (Di Mario et al., 2014b), introduces additional plasticity to the controller.

The eight range and bearing inputs (*rb_sect_k*) are obtained by dividing the bearing into eight sectors, and calculating the activation of each sector by taking the minimum range value measured in that sector and dividing it by the maximum possible range, which is 3.3 meters. The ninth input corresponds to the average of the headings among all the neighboring robots, in the robot's own coordinate system and normalized to the interval [-1,1]. The use of a single averaged input instead of one input per robot allows the controller to generalize to any number of robots.

Experimental Results and Discussion

The learning process is performed completely in simulation. We run three different optimization sets depending on

the learning schema and fitness function used: *global homogeneous* (centralized) learning, *local heterogeneous* (distributed) learning with *individual* movement factor, and *local heterogeneous* (distributed) learning with *group* movement factor. Since PSO is a stochastic optimization method, we perform 20 optimization runs for each of these learning schemes.

Each evaluation during the learning process has a duration of 45 s and takes place in an unbounded arena. Four robots are placed forming a square of side length equal to two robot diameters with random orientations. The local fitness function is calculated by the robots using only their internal measurements (simulated range and bearing and wheel encoders, both with added noise), while the global fitness function is calculated using the robots' global positions with no errors provided by the simulator.

The learning progress is shown in Fig. 5 for the three learning sets, representing the best solution found at each iteration for the three different learning approaches. The curves show the average of the 20 runs, and the error bars represent the standard deviation. In the case of local heterogeneous sets it shows not only the local metric but also the global one, since it is designed to reflect the quality of the flocking behavior and allows for comparison with centralized learning.

Comparing Fig. 5a with Fig. 5b and Fig. 5c, we can see that global homogeneous learning achieves the best global metric performance and lowest standard deviation of the three methods. Also, it requires less iterations to learn as the learning curve becomes flatter faster, although the homogeneous approach employs four times the evaluation time of the heterogeneous approaches for each iteration.

In Fig. 5c (heterogeneous with group movement factor), there is a perfect matching of local and global metrics. On the other hand, in Fig. 5b (heterogeneous with individual movement factor) the local and global metrics do not match initially, but they converge to the same value as the learning progresses.

The individual movement factor is easy to learn (robots just move straight), so it achieves a high value in the initial iterations. However, because of the lack of alignment and compactness, robots spread and do not achieve the desired behavior. As the run progresses, alignment and compactness factors improve, and therefore the difference between local and global metrics is reduced. Both alignment and compactness factors still have margin for learning and might produce an improvement with further iterations.

After the learning process is finished, the fitness of the best solution from each of the 20 independent learning runs is evaluated systematically in simulation, running 100 experiments of 60 s for each solution.

From Fig. 6a we can see that homogeneous learning achieves a high performance with low standard deviation for all the runs. Both heterogeneous approaches learn the

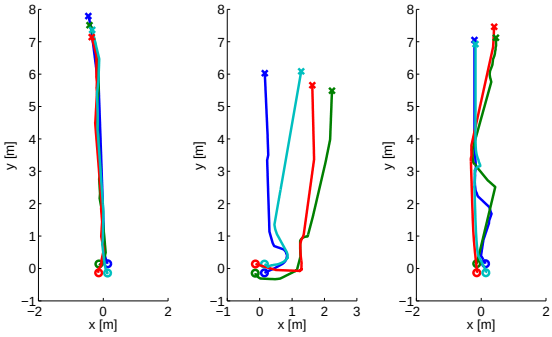


Figure 7: Example of trajectories of four robots flocking in simulation during 60 s for selected controllers from: (a) global homogeneous learning, (b) local heterogeneous with individual movement factor, and (c) local heterogeneous with group movement factor. The initial positions are marked with a circle, while the final positions are marked with a cross.

desired behavior in most runs, but sometimes fail, resulting in a high standard deviation. We noticed that this was due to two opposite reasons: in the heterogeneous learning with individual movement factor (Fig. 6b), individual speed is rewarded so the robots sometimes split. On the other hand, in heterogeneous with group movement factor (Fig. 6c) robots sometimes aggregate close to their initial positions in a very compact group and fail to travel far, resulting in low performance. This might be caused by obtaining very similar evaluations of the four particles tested together, which does not reflect the differences among the four controllers, discarding potential good solutions and promoting bad ones.

The difference in compactness of the group can be appreciated in the selected trajectories shown in Fig. 7, taken from the controller with highest median for each learning approach. In the case of the homogeneous controller, robots follow an almost perfect line. The trajectories followed by the heterogeneous controller learned with individual movement factor show that robots tend to spread, while those learned with group movement factor are more compact. These trajectories reflect the overall behaviors obtained by most of the 20 solutions of each learning approach.

In order to validate the results obtained in simulation, we select the controller with highest median for each learning approach and test it on real robots. We run 20 experiments for each solution. The initial positions and number of robots are the same as used for learning in simulation, but the evaluation time is reduced to 10 s in order to be able to keep track of the robots' positions during the whole evaluation due to the limited field of view of the overhead camera. Following the same scheme adopted in simulation, the local fitness function is computed on each robot using only its on-board resources, while the global fitness is computed externally

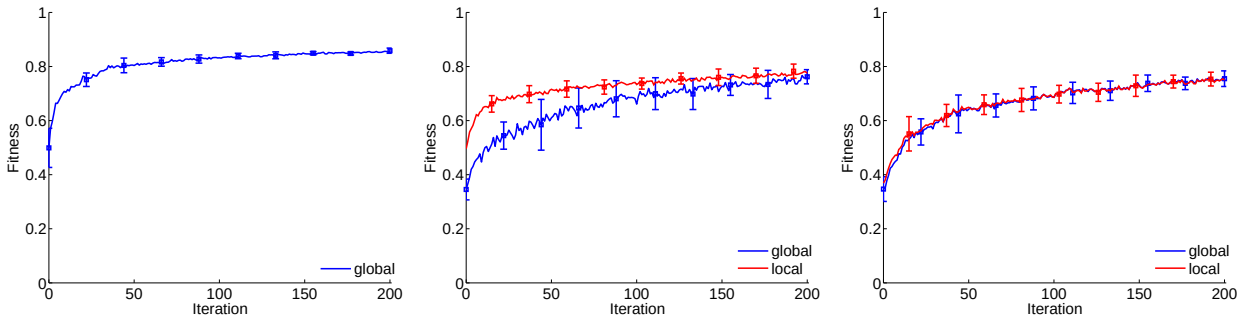


Figure 5: (a) Learning progress measured using the global metric for global homogeneous (centralized) learning . (b) Learning progress measured using the global (blue) and local (red) metrics for local heterogeneous (distributed) learning with individual movement factor. (c) Learning progress measured using the global (blue) and local (red) metrics for local heterogeneous (distributed) learning with group movement factor. The curves show the average of the 20 runs, and the error bars represent the standard deviation.

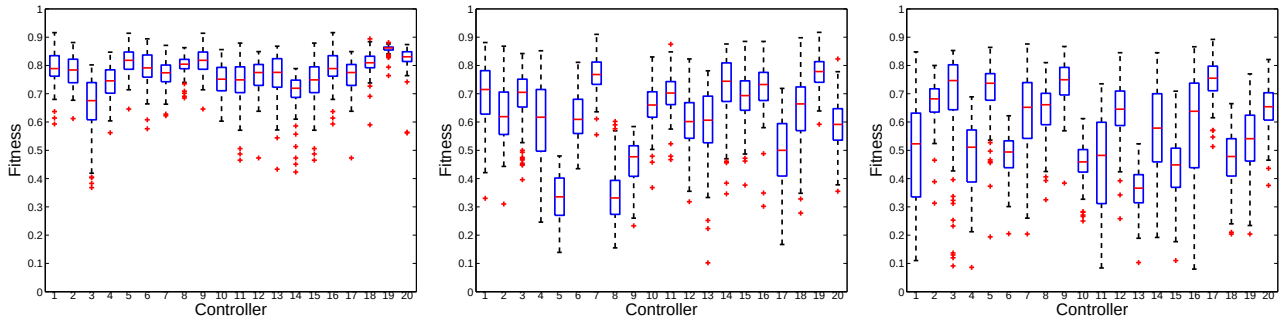


Figure 6: Performance measured with the global metric in simulation for the best solution found in each of the 20 independent learning runs with (a) global homogeneous (centralized) learning, (b) local heterogeneous (distributed) learning using individual movement factor and (c) local heterogeneous (distributed) learning using group movement factor. The box represents the upper and lower quartiles, the line across the middle marks the median, and the crosses show outliers for 100 evaluations of each controller.

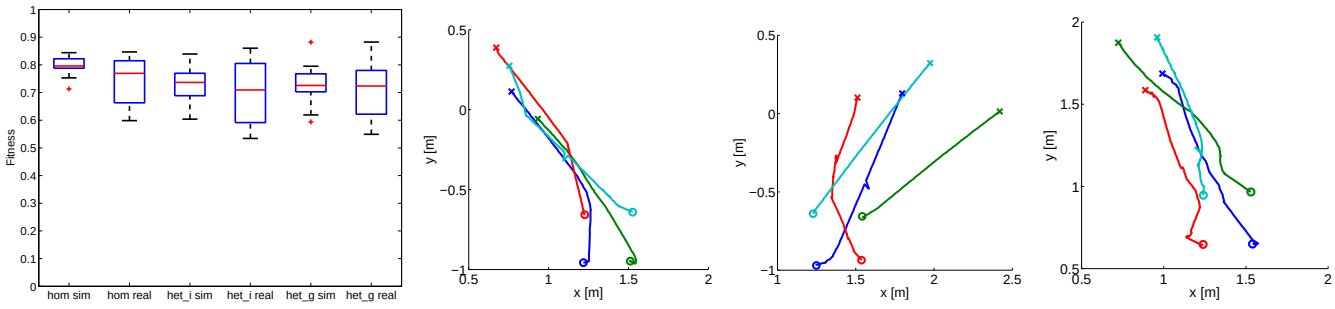
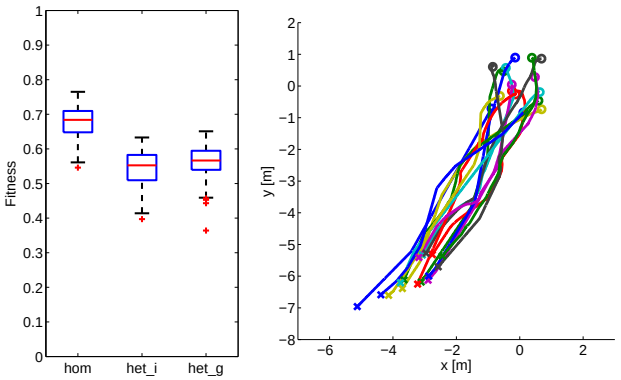


Figure 8: Evaluation with real robots for experiments of 10s for the selected controllers. (a) Performance measured for 20 evaluations per controller with the global metric with real robots and in simulation for comparison of the selected controllers for: global homogeneous learning (*hom real* and *hom sim*), local heterogeneous learning using individual movement factor (*het_i real* and *het_i sim*), and local heterogeneous learning using group movement factor (*het_g real* and *het_g sim*). Trajectories of a single experiment with real robots for: homogeneous (b), heterogeneous with individual movement factor (c), and heterogeneous with group movement factor (d). The initial positions are marked with a circle, the final positions are marked with a cross.



(a) (b)

Figure 9: Evaluation in simulation using 16 robots for experiments of 60 s. (a) Performance measured for 100 evaluations per controller using the global metric for: homogeneous learning (*hom*), heterogeneous learning using individual movement factor (*het_i*), and local heterogeneous learning using group movement factor (*het_g*). (b) Trajectories of a single experiment of homogeneous learning.

given the information provided by the overhead camera.

Fig. 8a shows the performance for the best controller from each learning approach both in simulation and reality. The shorter duration of experiments (10 s as opposed to 60 s) implies that the initial stage of aggregation and alignment of the robots represents a larger fraction of the total time, and therefore the performances for the 10 s runs are lower than for the 60 s runs. Both in simulation and reality the homogeneous controller achieves the highest median, but there is a higher variance in the results with real robots that suggests that some modeling details are missing in the simulation.

The trajectories observed on real robots in Fig. 8 show the same differences in compactness and spreading for the three approaches that were previously seen in simulation.

The last set of experiments that we conducted, in this case in simulation only, consisted in increasing the number of robots from four to 16 to see how the different controllers learned with four robots generalize to larger group numbers. Robots were initially positioned with a uniform random distribution on a squared area of 2 m side, and random orientations. Fig. 9a shows performance measured with the global metric in simulation.

As expected, the homogeneous controller generalizes quite well. Robots form one line or various close lines and move straight in a very compact group (see Fig. 9b). We did not have the same expectations for the heterogeneous controllers, given that robots might assume specialized roles. In fact, the behaviors observed were similar to the four robot case, but the robots tend to spread, especially in the case of the controller learned with individual movement factor. The compactness factor is also lower due to the fact that it is impossible for each robot to keep the same desired distance to every other robot in the group of 16, which results in lower performance values overall.

The resulting flocking behaviors can be better understood by looking at the video provided as supplementary material[1]. It shows experiments in simulation and with real robots for the three selected controllers and different number of robots.

Conclusion

We have seen that the fitness metrics used, based on Reynolds' rules, reflect an appropriate flocking behavior. They allowed us to obtain better and more robust solutions than in our previous work (Di Mario et al., 2014b) for both the centralized and the distributed learning. The use of the alignment factor helps to maintain the cohesiveness of the group, and also to match the local metric with individual movement factor with the global one. Additionally, we have shown that the learned controllers can generalize to groups of increasing number of robots, resulting in specially robust controllers in the homogeneous solution.

Our results show that the controllers learned in the homogeneous centralized approach have higher fitness and lower standard deviation than those learned in a distributed manner, although the centralized learning takes four times the evaluation time of the distributed strategies. Nevertheless, the best solutions found for centralized and distributed learning performed similarly, in simulation and in the different experiments with real robots.

In the case of the distributed learning with individual movement factor, group compactness and cohesiveness were reduced due to the prevalence of the individual movement while learning a collective task. Our solution to this issue was to define a new local metric with global movement factor, which mimics perfectly the global metric, yet it makes the distributed learning harder since four different particles evaluated together return very similar local fitness regardless of the difference in behaviors of the individual controllers (a typical credit assignment problem in distributed learning). Learning with homogeneous controllers using the local metric with group movement factor could be a way to bypass the credit assignment problem while still allowing for on-board learning with simple noisy sensors without the need of external hardware, but it would not decrease the evaluation time per iteration through parallel evaluations.

As continuation of this work, we will explore new strategies for heterogeneous distributed learning that might result in better and more consistent performances of collective tasks. We will explore other neighboring topologies of the PSO algorithm, as well as different ways of distributing the particles among the robots. In addition, we would like to explore learning in the presence of obstacles in order to generate obstacle avoidance at the group level.

[1] http://disalw3.epfl.ch/research/distributed_adaptation/ecal.mp4

Acknowledgements

This research was supported by the Swiss National Science Foundation through the National Centre of Competence in Research Robotics.

References

Antonelli, G., Arrichiello, F., and Chiaverini, S. (2008). Flocking for multi-robot systems via the null-space-based behavioral control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1409–1414.

Balch, T. and Arkin, R. (1998). Behavior-based formation control for multi-robot teams. *IEEE Transactions on Robotics and Automation*, 14(6):926–939.

Baldassarre, G., Trianni, V., Bonani, M., Mondada, F., Dorigo, M., and Nolfi, S. (2007). Self-organized coordinated motion in groups of physically connected robots. *IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics*, 37(1):224–39.

Celikkanat, H. (2008). Optimization of self-organized flocking of a robot swarm via evolutionary strategies. In *International Symposium on Computer and Information Sciences*.

Chang, Y.-H., Chen, C.-L., Chan, W.-S., Lin, H.-W., and Chang, C.-W. (2013). Fuzzy formation control and collision avoidance for multiagent systems. *Mathematical Problems in Engineering*, volume 2013.

Di Mario, E. and Martinoli, A. (2014a). Distributed particle swarm optimization for limited time adaptation in autonomous robots. In *Eleventh Int. Symp. on Distributed Autonomous Robotic Systems, Springer Tracts in Advanced Robotics*, volume 104, pages 383–396.

Di Mario, E. and Martinoli, A. (2014b). Distributed particle swarm optimization for limited time adaptation with real robots. *Robotica*, 32(02):193–208.

Di Mario, E., Navarro, I., and Martinoli, A. (2014a). Analysis of fitness noise in particle swarm optimization: From robotic learning to benchmark functions. In *IEEE Congress on Evolutionary Computation*, pages 2785–2792.

Di Mario, E., Navarro, I., and Martinoli, A. (2014b). Distributed learning of cooperative robotic behaviors using particle swarm optimization. In *14th International Symposium on Experimental Robotics*, to appear in *Springer Tracts in Advanced Robotics*.

Etemadi, S., Vatankhah, R., Alasty, A., Vossoughi, G., and Boroushaki, M. (2012). Leader connectivity management and flocking velocity optimization using the particle swarm optimization method. *Scientia Iranica*, 19(5):1251 – 1257.

Floreano, D. and Mondada, F. (1996). Evolution of homing navigation in a real mobile robot. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 26(3):396–407.

Gauci, M., Chen, J., Dodd, T., and Groß, R. (2014). Evolving aggregation behaviors in multi-robot systems with binary sensors. In *Eleventh Int. Symp. on Distributed Autonomous Robotic Systems, Springer Tracts in Advanced Robotics*, volume 104, pages 355–367.

Hayes, A. T. and Dormiani-Tabatabaei, P. (2002). Self-organized flocking with agent failure: Off-line optimization and demonstration with real robots. In *IEEE Int. Conf. on Robotics and Automation*, pages 3900–3905.

Jin, Y. and Branke, J. (2005). Evolutionary optimization in uncertain environments: A survey. *IEEE Transactions on Evolutionary Computation*, 9(3):303–317.

Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. In *IEEE International Conference on Neural Networks*, pages 1942 – 1948.

Lee, S.-M. and Myung, H. (2013). Particle swarm optimization-based distributed control scheme for flocking robots. In *Robot Intelligence Technology and Applications*, volume 208, pages 517–524.

Lochmatter, T., Roduit, P., Cianci, C., Correll, N., Jacot, J., and Martinoli, A. (2008). SwisTrack - a flexible open source tracking software for multi-agent systems. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4004–4010.

Matarić, M. (2001). Learning in behavior-based multi-robot systems: Policies, models, and other agents. *Cognitive Systems Research*, 2:81–93.

Michel, O. (2004). Webots: Professional mobile robot simulation. *Advanced Robotic Systems*, 1(1):39–42.

Morihiro, K., Isokawa, T., Nishimura, H., and Matsui, N. (2006). Emergence of flocking behavior based on reinforcement learning. In *Knowledge-Based Intelligent Information and Engineering Systems*, volume 4253, pages 699–706.

Navarro, I. and Matía, F. (2011). A framework for collective movement of mobile robots based on distributed decisions. *Robotics and Autonomous Systems*, 59(10):685–697.

Olfati-Saber, R. (2006). Flocking for multi-agent dynamic systems: Algorithms and theory. *IEEE Transactions on Automatic Control*, 51:401–420.

Parker, L. E. (1997). L-ALLIANCE : Task-oriented multi-robot learning in behavior-based systems. In *Advanced Robotics, Special Issue on Selected Papers from IROS'96*, pages 305–322.

Pugh, J. and Martinoli, A. (2009). Distributed scalable multi-robot learning using particle swarm optimization. *Swarm Intelligence*, 3(3):203–222.

Pugh, J., Raemy, X., Favre, C., Falconi, R., and Martinoli, A. (2009). A fast on-board relative positioning module for multi-robot systems. *Special issue on Mechatronics in Multi-Robot Systems, IEEE Trans. on Mechatronics*, 14(2):151–162.

Pugh, J., Zhang, Y., and Martinoli, A. (2005). Particle swarm optimization for unsupervised robotic learning. In *IEEE Swarm Intelligence Symposium*, pages 92–99.

Reynolds, C. W. (1987). Flocks, herds, and schools: A distributed behavioral model. *Computer Graphics*, 21(4):25–34.

Zhang, Y., Antonsson, E., and Martinoli, A. (2008). Evolutionary engineering design synthesis of on-board traffic monitoring sensors. *Research in Engineering Design*, 19(2):113–125.